Complexity Links Between Matrix Multiplication, Klee's Measure and Call Access Control for Satellite Constellations

Jérôme Galtier, Paolo Penna

▶ **To cite this version:**

Jérôme Galtier, Paolo Penna. Complexity Links Between Matrix Multiplication, Klee's Measure and Call Access Control for Satellite Constellations. RR-4166, INRIA. 2001. inria-00072456

HAL Id: inria-00072456
https://inria.hal.science/inria-00072456

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Complexity Links Between Matrix Multiplication, Klee's Measure and Call Access Control for Satellite Constellations

Jérôme Galtier — Paolo Penna

N° 4166

Avril 2001

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4166--FR+ENG

INRIA
SOPHIA ANTIPOLIS

Complexity Links Between Matrix Multiplication, Klee's Measure and Call Access Control for Satellite Constellations

Jérôme Galtier*, Paolo Penna

Thème 1 — Réseaux et systèmes
Projet Mascotte

Rapport de recherche n° 4166 — Avril 2001 — 19 pages

Abstract: We study the complexity of computing the maximum load of weighted rectangles (the load on a point is defined as the sum of the weights of intersecting rectangles on this point). We relate the complexity of the dynamic version of this problem (in which we want to perform insert/delete operations on the rectangles) to the complexity of the matrix multiplication problem in the $(\max, +)$ algebra. In particular, we show that a significant improvement on the existing solutions for the first one (within $O(n^{\theta/2})$ per insertion/extraction with $\theta < 1$) implies an improvement on the complexity of the latter one (to $O(N^{2+\theta})$). Furthermore, this connection makes possible to state a lower bound on a general class of algorithms that include all the most efficient previously known algorithms for the Klee's measure problem. This implies that such solutions cannot be improved by more than a logarithmic factor in their actual form. Finally, all the above results also apply to variations/restrictions of the *depth* problem motivated by LEO satellite constellations routing problems.

Key-words: Computational geometry, dynamic algorithms, algorithms for mobile and net computing, computational complexity.

* also with France Télécom R&D

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Relations de complexité entre la multiplication de matrices, la mesure de Klee et le contrôle d'accès pour les constellations de satellites

Résumé : Nous étudions la complexité du calcul de la charge maximum d'un ensemble de rectangles munis d'un poids (la charge en un point étant la somme des poids des rectangles s'intersectant en ce point). Nous relions la complexité de la version dynamique de ce problème (dans laquelle on s'autorise des insertions/deletions de rectangles) à la complexité du problème de la multiplication de matrices dans l'algèbre $(\max, +)$. En particulier, nous montrons qu'une amélioration significative de la solution de la première (en $O(n^{\theta/2})$ par insertion/extraction avec $\theta < 1$) implique une amélioration de la complexité de la seconde (en $O(N^{2+\theta})$). De plus, cette relation rend possible la formulation d'une borne inférieure pour une classe générale d'algorithmes contenant les algorithmes les plus efficaces connus pour la résolution du problème de la mesure de Klee. Cela implique que ces algorithmes ne peuvent être améliorés dans leur forme actuelle d'un facteur de plus de $\log(n)$. Enfin, les résultats obtenus s'appliquent aussi à des variantes du problème de la profondeur motivé par le routage dans les constellations de satellites.

Mots-clés : Géometrie algorithmique, algorithmes temps-réel, algorithmes pour la mobilité et les réseaux, complexité

Complexity Links Between Matrix Multiplication, Klee's Measure and Call Access Control for Satellite Constellations

Jérôme Galtier , Paolo Penna

e study the complexity of computing the maximum load of weighted rectangles (defined as the sum of the weights of intersecting rectangles on a single point). We relate the complexity of the dynamic version of this problem (in which we want to perform insert/delete operations on the rectangles) to the complexity of the matrix multiplication problem in the $(\max, +)$ algebra. In particular, we show that a significant improvement on the existing solutions for our problems implies an improvement on the complexity of the latter one. Furthermore, this connection makes possible to state a lower bound on a general class of algorithms that include all the previously known solutions. This implies that such solutions cannot be improved by more than a logarithmic factor. Finally, all the above results also apply to variations/restrictions of the *depth* problem motivated by LEO satellite constellations routing problems.

Keywords. Computational geometry, dynamic algorithms, algorithms for mobile and net computing, computational complexity.

1 Introduction

In the last years, a set of LEO satellite constellations has been launched or proposed as revolutionary communication systems. LEO satellites are orbiting at constant speed and constant altitude fixed between 500km and 2000km. As a consequence, the covering cell of a satellite moves continuously. So, in order to provide global coverage, we need several LEO satellites describing a set of regularly spaced orbits. The most popular examples of these systems are probably Globalstar, with 48 satellites on 8 orbits, and Iridium, with 6 polar orbits of 11 satellites each. Communications links provided by these constellations are bi-directional, and of type satellite-phone, satellite-gateway, and in case among the satellites (we then refer to inter-satellite links, or ISLs), which allows routing inside the satellite network, *without transiting through terrestrial networks.*

As a consequence, even a simple permanent connection between two fixed users turns out to be an interesting problem. The term connection here refers to a demand given by

*also with France Télécom R&D

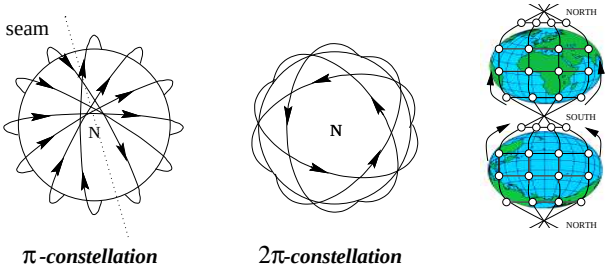

Figure 1: Two types of investigated constellations

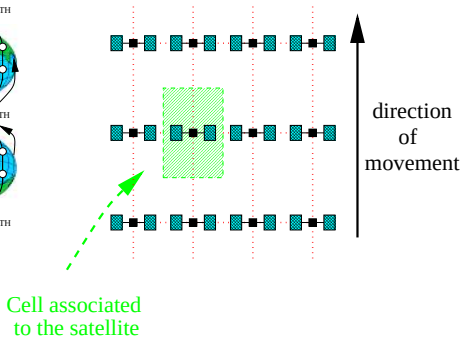

Figure 2: A simple model.

two positions, and some rate. This can be a phone call or more realistically some ATM-like request. Given a set of such requests, we would like to *guarantee resources to the users whatever the position of the satellites is (guaranteed hand-over).*

In order to isolate the kernel problem of hand-overs and routing in satellite constellations, we consider only circular orbits, we can suppose that a satellite describes at constant speed his orbit, a circle centered on the Earth. A basic brick for global coverage consists in a series of satellites regularly spaced on the same orbit. Once this belt is obtained, two policies are generally adopted. The first one, the π-constellation, consists in deploying these belts along a semi-circle when viewed from a pole, as shown in Figure 1. The second, the 2π-constellation, consists in spacing the orbits along a complete circle.

Usually, π-constellations use polar orbits (i.e. orbits that "roughly" cross the polar axis) for coverage reasons, and therefore are called "polar" constellations. Similarly, inclined orbits allow to make a better optimization of 2π-constellations, hence the name "inclined" constellations. However, there is no technical reason to forbid the use of polar orbits on inclined constellations and reversely. This is why we rather use the terminology $\pi/2\pi$-constellations which is less confusing.

The π-constellation is the structure of the Iridium system [13, 9] and at the basis of the original plans for Teledesic [16, 12]. A 2π-constellation forms the shape of the Globalstar service [4], and has also been planned for the future Skybridge project and the abandoned Celestri.

For a better comprehension of the model derived, we restrict ourselves to polar circular orbits at a given altitude. We space regularly n_o orbits in a π- or 2π-constellation, and each orbit contains n_s satellites regularly spaced. Each satellite communicates with his two nearest neighbors within its orbit, and the two satellites that have the same latitude in the two nearest neighboring orbits. We obtain likewise square-shaped cells, as illustrated in Figure 2 for a π-constellation. The 2π version of it implies that each part of the earth is covered both by an ascending and a descending satellite. This model, if not fully cost-

effective, remains realistic. Something closer to the final (industrial) proposition would be obtained by switching off the ISLs when a satellite-endpoint crosses one pole, and interleaving satellites in neighboring orbits so as to obtain hexagonal cells.

In the following, we consider a seamless simple-coverage constellation. This assumption is true in a π-constellation when the communications are short enough to end before they are interrupted by the crossing of the seam. Also it holds on a permanent basis in a 2π-constellation where we choose to cover a terminal uniquely by ascending (or descending) satellites.

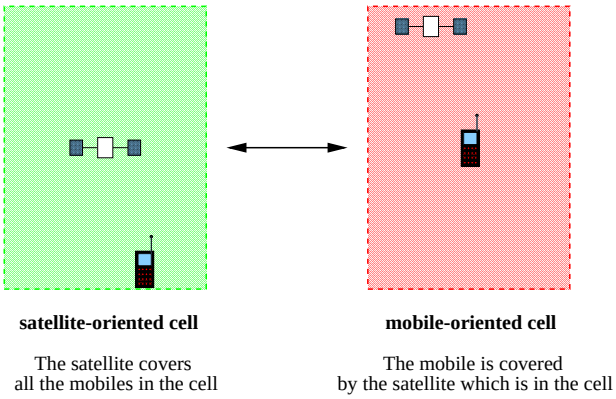

Figure 3: Dual footprint of a terminal in the sky.

According to the model in [8], in order to guarantee sufficient resources to an (active) user, we mainly exploit the following two concepts:

- We consider a dual notion to the satellite footprint, that is the terminal "footprint" in the sky, which is a conceptual terrestrial cell assigned to the terminal that contains exactly one satellite at any moment of time, as illustrated in Figure 3. If A is the terminal, we note $sat(A)$ such a footprint.
- Concerning the routing problem between two satellites A and B, we always choose the (unique) shortest path. In particular, denote B the terminal with the higher altitude (i.e. the closest to one pole); the route then consists of inter-orbital links between A and an intermediate satellite AB, and then intra-orbital links from AB to B (see Figure 4). This route is the shortest in number of links as well as in delay.

The main idea of this model is the following: if two (or more) terminals are "close", then their footprints intersect. This means that when a satellite occurs in such intersection, such a satellite is *the only one* that can serve those terminals. In other words, the maximum number of overlapping mobile-cells corresponds to the worst case number of terminals that must be directly connected to a same satellite (see Th. 2 below).

From this model, we can also characterize the position of the intermediate satellite $sat(AB)$, which is in a square cell, whose bound latitudes are the ones of $sat(A)$ and bound longitudes the ones of $sat(B)$, as shown in Figure 4. Note that there is always exactly one satellite in the cell, so that the position of the satellite is uniquely determined. Taking more advantage of the model, we can also localize the position of the reserved inter-orbital links from $sat(A)$ to $sat(AB)$. To each link l, we assign a center c_l located in the middle of it, as illustrated in the upper part of Figure 5. Then the centers of the links are contained in a rectangle $link(AAB)$ whose higher and lower latitudes are the ones of $sat(A)$, and whose vertical sides intersect respectively the center of $sat(A)$ and the center of $sat(AB)$. Once again, whatever the exact position of the satellite is, this definition gives all the links that are needed to connect the satellite of $sat(A)$ to the one of $sat(AB)$ and only them. Similarly, $rect(ABB)$ defines all the intra-orbital links that are required to connect $sat(AB)$ to $sat(B)$.

The following two results then hold:

Theorem 1 (Link Reservation [8]) *Given a set of communication requests $\mathcal{S} = ((A_i, B_i))_{1 \leq i \leq n}$, where B_p is the closest of $\{A_p, B_p\}$ to one of the poles, then a shortest-delay routing in a seamless simple-coverage constellation for all the request can be done if and only if the capacities C_{inter} of inter-orbital links and C_{intra} of intra-orbital links verify:*

$$C_{inter} \geq \max \left\{ |I| \text{ such that } I \subseteq \{1, \ldots, p\} \text{ and } \bigcap_{i \in I} rect(A_i A_i B_i) \neq \emptyset \right\} \quad (1)$$

$$C_{intra} \geq \max \left\{ |I| \text{ such that } I \subseteq \{1, \ldots, p\} \text{ and } \bigcap_{i \in I} rect(A_i B_i B_i) \neq \emptyset \right\} \quad (2)$$

Similarly, an *access* problem can be defined from the previous considerations, and we can state the following theorem :

Theorem 2 (Access Reservation [8]) *Given a set of communication requests $\mathcal{S} = (A_i, d_i)_{i \in \{1, \ldots, p\}}$, then we can guarantee resources on the closest satellite for all the communications in the system if and only if the maximum capacity C of a satellite verifies :*

$$C \geq \max \left\{ \sum_{i \in I} d_i \text{ such that } I \subseteq \{1, \ldots, p\} \text{ and } \bigcap_{i \in I} sat(A_i) \neq \emptyset \right\} \quad (3)$$

1.1 Our Results

In this work we extensively study the computational complexity of the *access* reservation and *link* reservation problems in LEO constellations, according to the previously described model and to Th.s 1-2. Because of the telecommunication environment, two different approaches can be considered. The *static* approach addresses the question of feasibility *once all the users are known.* In the *dynamic* one, in a call access perspective, we ask whether an incoming call is acceptable while others are already present. This is a typical situation arising in

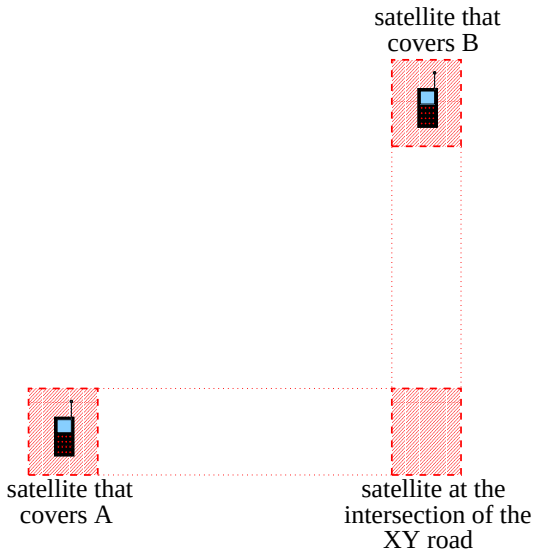

Figure 4: Position of the intermediate satellite in a shortest-delay XY route.

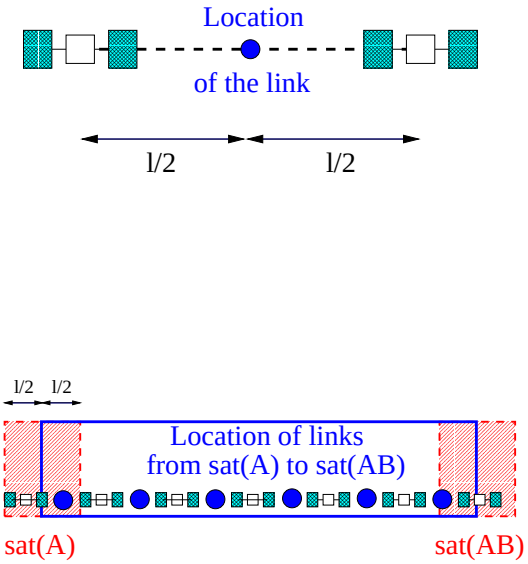

Figure 5: Position of the inter-orbital links in a shortest-delay XY route.

dynamic algorithms where, given an update (of a certain kind) of the input, one wants to find an optimal solution possibly without recomputing everything from scratch. Usually, one aims to build a data structure that supports updates/queries efficiently. So, the complexity measure now is the worst case time complexity needed to perform updates/queries.

Our results can be summarized as follows:

- We provide several reductions among our problems and other classical problems arising in computational geometry, thus showing that all such problems are essentially the same;
- Then, we look closer at the dynamic approach and we show that any data structure (for the access or for the link problems) requires $\Omega(\sqrt{n})$-time per update/query, unless the matrix multiplication in the $(\max, +)$ algebra can be solved within $O(N^\theta)$-time, for some $\theta < 3$;
- By exploiting a lower bound on the complexity (over the fixed programs model of computation) of the latter problem, we derive tight bounds on the complexity (on a general class of algorithms) of all the dynamic versions of our problems.

The latter two results give a strong evidence that computing several (say $O(\sqrt{n})$) updates/queries is computationally more expensive than computing a single global solution (the static version of the problem).

It is worth to observe that up to now no $O(N^\theta)$-time algorithm, for some $\theta < 3$, for matrix multiplication in $(\max, +)$ is known (except when the numbers in the matrices are bounded so that we can use the classical matrix-matrix multiplication). A closely related problem is the all-pairs shortest path on directed weighted graphs (it is easy to see that this is equivalent to matrix multiplication in the $(\min, +)$ algebra). The best known algorithm for this problem has running time just barely sub-cubic (namely $O(N^3(\log\log N/\log N)^{1/3})$ in [5] improved to $O(N^3(\log\log N/\log N)^{1/2})$ in [17]), while better solutions exist only when the weights are bounded above [18, 19] or for undirected graphs [6, 7, 15].

Further aspects concern the connections that we made with other computational geometry problems. Indeed, all the problems are equivalent to the problem of computing the maximum (weighted) clique of an intersection graph (see [10] and Problem 9 below), and thus all the results also apply to that problem.

Additionally, the upper bounds for the "dynamic" version of all the problems derive from (a modification of) previous solutions for the Klee's measure problem (see [14, 2] and Problem 8 below). So, if a more efficient dynamic algorithm for the latter exists, then it should not work for the other problems (as all the existent solutions instead do).

Organization of the paper. In Section 1.2 we provide some basic definitions. The connection with the matrix multiplication problem is given in Section 2 together with some tight bounds on the complexity of these problems. In Section 3 we describe the problems arising from the satellites constellation model and the other computational geometry problems, while in Section 3.3 we relate all such problems. Finally, in Section 4 we conclude and outline some open problems. For the sake of completeness some proofs are provided in the Appendix.

1.2 Preliminaries

Let $v_1, \ldots, v_d$ be a series of independent vectors in the d-dimensional Euclidean space $\mathbb{R}^d$. We say that an hypercube R is *regular* with respect to $v_1, \ldots, v_d$, if one can find $a_1, \ldots, a_d$ and $b_1, \ldots, b_d$ such that

$$R = \{x \in \mathbb{R}^d \quad / \quad \forall i \in \{1, \ldots, d\} \quad a_i \leq x \cdot v_i < b_i\}.$$

Given an instance of a problem PROB we distinguish a part of the input called *weights*. A *funny algorithm*[1] is an algorithm which uses the usual loop instructions, conditional instructions and variable assignment instructions, but can apply on the weights only the 'max' and '+' operators. We consider furthermore that any result of the 'max' and '+' operations involving at least one weight is also a weight. Finally, no weight occurs in the loop instructions (e.g. instructions like '`For` $i := 1$ `To` $\max(\cdot)$' are not allowed) nor in a predicate p used by some conditional instructions (e.g. '`If` p `Then` Instr1 `Else` Instr2').

A *fixed program* is a finite sequence of instruction of the form

$$t_i := \alpha \oplus \beta,$$

[1] The name refers to the "funny algebra" term used for the $(\max, +)$ algebra.

where $\oplus$ is some instruction repertoire $\mathcal{I}$ available to the program, α and β are either input variables of the program, constants, or previous t_j's (i.e. $j < i$). Moreover, we say that the program computes a set of functions $\{f_k\}$ if, for every k, the value of some t_i is equal to f_k, for all the possible arguments of the function. The *complexity for fixed programs* over $\mathcal{I}$ of a set of functions $\{f_k\}$ is the length of the shortest fixed program, with instruction repertoire $\mathcal{I}$, computing $\{f_k\}$. In the sequel we will restrict to fixed programs over $\mathcal{I} = \{\max, +\}$.

Given a problem PROB, we denote by $M - \text{PROB}$ the restriction of PROB in which the weights belongs to the set $\{1, 2, \ldots, M\}$. Finally, $\# - \text{PROB}$, denotes the variation of the problem PROB in which we require the solution to be computed by some funny algorithm. Given a problem PROB and another problem OTHER, we denote by $\text{PROB} \leq \text{OTHER}$ the fact that PROB is linear-time reducible to OTHER. Additionally, $\# - \text{PROB} \leq \# - \text{OTHER}$ means that the reduction is computable in linear time by *some funny algorithm*. If we have simultaneously $\text{PROB} \leq \text{OTHER}$ and $\text{OTHER} \leq \text{PROB}$, then we say that $\text{PROB} \sim \text{OTHER}$. If an algorithm with time complexity $O(f(n))$ exits for problem PROB, where f is an increasing function with n, then we write $\text{PROB} \leq f(n)$. On the other hand, if any algorithm which solves PROB has time complexity $\Omega(g(n))$, where g is an increasing function with n, then we write $\text{PROB} \geq g(n)$. Thus, by using the '$\leq$' notation, $\# - \text{PROB} \geq f(n)$ refers to a lower bound on the complexity of funny algorithms that solve PROB. Finally, $\text{PROB} \leq \text{OTHER}$ always denote the fact that both $\# - \text{PROB} \leq \# - \text{OTHER}$ and $M - \text{PROB} \leq (cM) - \text{OTHER}$, for some constant c, hold. The '$\sim$' symbol is used in the same way. Finally, it is easy to see that '$\leq$' satisfies the transitive property.

2 The Core of the Problem(s): Matrix-Matrix Multiplication

***Problem* 1** (DYNAMIC DEPTH^d) *Let $v_1, \ldots, v_d$ be a series of independent vectors in $\mathbb{R}^d$, and $R^1, \ldots, R^{n+1}$ be a series of regular hypercubes with respect to $v_1, \ldots, v_d$. Let $w_1, \ldots, w_{n+1}$ be weights assigned to the rectangles, and a data structure $\mathcal{D}(\mathcal{S})$ associated to $\mathcal{S} = ((R^1, w_1), \ldots, (R^n, w_n))$, compute either*

- *an update of $\mathcal{D}(\mathcal{S})$ by adding (R_{n+1}, w_{n+1}),*
- *an update of $\mathcal{D}(\mathcal{S})$ by withdrawing (R_n, w_n),*
- *the maximum load $\sum_{j \in I} w_j$ for a set $I \subseteq \{1, \ldots, n\}$ such that $\bigcap_{j \in I} R^j \neq \emptyset$.*

The complexity of the problem is given by the maximum complexity of the three operations, multiplied per n.

***Problem* 2** ($(\max, +)-$MULTIPLICATION **(with $n = N^2$)**) *Given two $N \times N$ square matrices $A = (a_{i,j}), 1 \leq i, j \leq N$ and $B = (b_{i,j}), 1 \leq i, j \leq N$. , compute, for each $1 \leq i, j \leq N$*

$$c_{i,j} := \max_{1 \leq k \leq N} a_{i,k} + b_{k,j}.$$

The weights of the problem are the entries of the two matrices.

The main result of this section is the following result relating the dynamic version of the DEPTH^2 problem with the complexity of matrix multiplication in $(\max, +)$.

Theorem 3 $(\max, +)-$MULTIPLICATION $\leq$ DYNAMIC DEPTH^2.

PROOF. Consider two $N \times N$ matrices A and B. We define an instance of DYNAMIC DEPTH of size $n = N^2$ as follows:

Encoding of A: Consider a square grid of side N and for each cell at row i and column k, add a square matching the cell and having weight $a_{i,k}$ (See Figure 6(a)).

Encoding of B: Let be $A_i = (a_{i,1}, \ldots, a_{i,N})$ and $B^j = (b_{1,j}, \ldots, b_{N,j})$. We will show how to encode, for every j, with $1 \leq j \leq N$, the N vector-vector products

$$\max(A_1 + B^j), \max(A_2 + B^j), \ldots, \max(A_N + B^j) \tag{4}$$

using only $4N$ insert/delete operations of new rectangles. In particular, we first add N rectangles $R_1^j, \ldots, R_n^j$ encoding B^j. This set of rectangles is shown in Figure 6(b): They matches the columns of the grid and their weights are $b_{1,j}, \ldots, b_{N,j}$, respectively. Then, for each row i, we add (and just after withdraw) an "horizontal" rectangle $\overline{R_i}$ covering the i-th row of the grid and having weight $L_{high} = \max_{i,j}(a_{i,j}) + \max_{i,j}(b_{i,j})$.

Consider the generic step in which we added the rectangle $\overline{R_i}$. It is easy to see that the maximum depth of this arrangement is reached on the i-th row and its value $L_{i,j}$ is given by

$$L_{i,j} = L_{high} + \max_{1 \leq k \leq N} a_{i,k} + b_{k,j}. \tag{5}$$

From the above equation it follows that the vector-vector product $\max(A_i + B^j)$ can be easily computed by adding $\overline{R_i}$ and checking the (maximum) depth value $L_{i,j}$. So, by adding/removing the N rectangles $\overline{R_1}, \ldots, \overline{R_N}$, we can compute the N vector-vector products of Eq. 4. Finally, we can iterate this construction by first removing all the rectangles encoding B^j and then restarting with a new set of N rectangles encoding B^{j+1}. Thus, the overall number of add/withdraw (and load queries) necessary to compute $\max(A + B)$ is $4N^2$. □

Corollary 1 *If a $\theta < 1$ exists such that* DYNAMIC DEPTH^2 *can be solved within $O(n^{3\theta/2})$-time, then* $(\max, +)-$MULTIPLICATION *can be solved within $O(N^{3\theta}) = O(n^{3\theta/2})$-time.*

The above result implies that a significant improvement of the algorithms in [14, 2] would also imply an improvement for both matrix-matrix multiplication in the $(\max, +)$ algebra and for the all-pairs shortest path problems, a two long-standing open problems.

Several solutions proposed for the Klee's measure problem can also be adapted to our problem. In particular:

- In [10], an $O(n \log(n))$-time algorithm is given to find the maximum intersection of n rectangles in two dimensions.

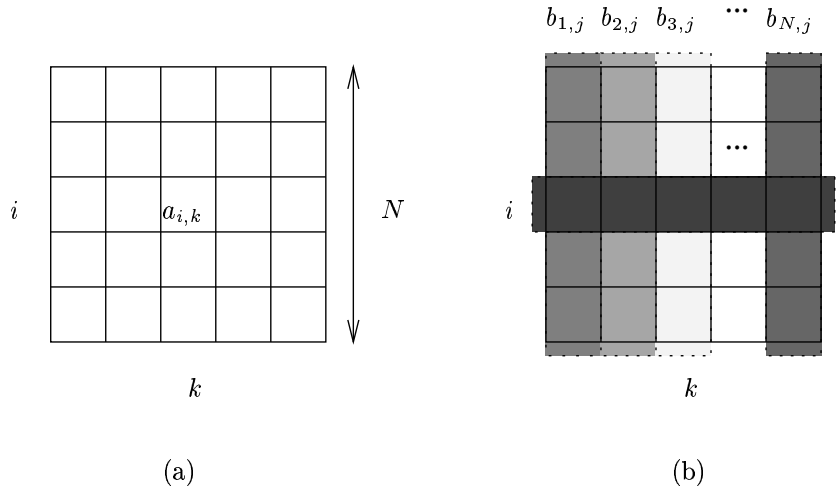

Figure 6: The reduction from $(\max,+)-$MULTIPLICATION to DYNAMIC DEPTH: (a) the static part representing the matrix A; (b) the set of rectangles representing $\max(A_i + B^j)$.

- In the d-dimensional case, the best known algorithm takes $O(n^{d/2}\log(n))$ time [14].
- The same complexity, $O(n^{d/2}\log(n))$ is obtained with a *quadtree* structure for small dimensions in [2] (the proof is given only for dimensions three and four).

Indeed, the latter two solutions also apply to the DYNAMIC DEPTH^{d-1} problem (see Proposition 1 below), thus yielding an $O(n^{3/2})$-time algorithm for DYNAMIC DEPTH^2. Some interesting remarks are done in [14, 2] on the optimality of their algorithm, more precisely, "it might be possible to shave of the factor of $\log(n)$. But in fact, there is no reason to believe that the method is even near optimal". In the next section, will show that the complexity (for fixed programs) of the $\#-$DYNAMIC DEPTH (and thus also of $\#-$SAT CAC LINKS and $\#-$SAT CAC ACCESS, as we will see below) is greater than $n^{3/2}$, which en-lights a useful context where the algorithm is almost optimal.

Although finding general lower bounds for such problem is a per se huge problem in computational complexity, we can state some lower bounds on the complexity of a certain (general) class of algorithms that solve the problem: the funny algorithms. Such a class include all the existing solutions, which implies that a more efficient solution may be achieved only through an approach of different type.

To this aim we will use the concept of fixed programs complexity.

2.1 Tight Bounds on the Complexity

In order to capture the complexity of funny programs of our problems, we need to consider fixed programs which are "specialized" on some fixed set of hypercubes. In other words, now *the shapes* of the hypercubes are fixed and the input is given by their weights only:

***Problem* 3** ($\#-$DYNAMIC DEPTH$^{d,\mathcal{R}}$) *Let $v_1,\ldots,v_d \in V$ be a series of independent vectors in $\mathbb{R}^d$, and $\mathcal{R}=\{R^1,\ldots,R^n\}$ be a series of regular* fixed *hypercubes with respect to $v_1,\ldots,v_d$. We define the $\#-$*DYNAMIC DEPTH$^{d,\mathcal{R}}$ *as the $\#-$*DYNAMIC DEPTH^{d} *problem restricted to hypercubes from the set $\mathcal{R}$, in which the output function is the depth value after each insert/withdraw operation.*

The following theorem provides a useful connection between the class of funny algorithms and the fixed program complexity (the proof is given in the Appendix).

Theorem 4 *If a funny algorithm* Alg *solving the* DYNAMIC DEPTH^{d} *has time complexity $f(n)$, then, for every $\mathcal{R}$, there exists a fixed program for $\#-$*DYNAMIC DEPTH$^{d,\mathcal{R}}$ *having complexity $O(f(n))$.*

The main brick to prove lower bounds on the complexity will be the following result:

Theorem 5 ([11], Chapter 6, Th. 6.1 & pag. 57) *The complexity of fixed programs for the*
$(\max,+)-$MULTIPLICATION *problem is bounded below by $N^3=n^{3/2}$.*

By combining Th. 3, Th. 4 and Th. 5 we obtain the following almost tight bounds (the upper bound follows from [14, 2]):

Corollary 2 $n^{3/2} \leq \#-$DYNAMIC DEPTH$^2 \leq n^{3/2}\log n$.

3 Application to the SAT Problems

In this section, we establish the connexion between our complexity issues and satellite communication problems. We justify in this way the "dynamic" version of the maximum depth problem that was studied until now.

3.1 Satellite Constellations and Restriction to Fixed Sides

According to Th.s 1-2 we have to face two different problems of intersection of rectangles. In particular, the "link" problem uses rectangles of *fixed width*, but *variable height*. Meanwhile, the "access" problem uses *both fixed width and height*.

For each of these two problem we can consider two versions: the *static* and the *dynamic* one, as discussed in Section 1.1. We can therefore define the following problems :

***Problem* 4** (SAT TOT LINKS) *Given a set of communication requests* $\mathcal{S} = (A_i, B_i, d_i)_{i\in\{1,\ldots,n\}}$, *where* B_i *is the closest of* $\{A_i, B_i\}$ *to one of the poles, and* d_i *is a weight, compute*

$$\max\left\{\sum_{i\in I} d_i \text{ such that } I \subseteq \{1,\ldots,n\} \text{ and } \bigcap_{i\in I} rect(A_iA_iB_i) \neq \emptyset\right\}$$

***Problem* 5** (SAT TOT ACCESS) *Given a set of communication requests* $\mathcal{S} = (A_i, d_i)_{i\in\{1,\ldots,n\}}$, d_i *being a weight, compute*

$$\max\left\{\sum_{i\in I} d_i \text{ such that } I \subseteq \{1,\ldots,n\} \text{ and } \bigcap_{i\in I} sat(A_i) \neq \emptyset\right\}$$

***Problem* 6** (SAT CAC LINKS) *Given a set of communication requests* $\mathcal{S}' = (A_i, B_i, d_i)_{i\in\{1,\ldots,n+1\}}$, *where* B_i *is the closest of* $\{A_i, B_i\}$ *to one of the poles,* d_i *is a weight, and a data structure* $\mathcal{D}(\mathcal{S})$ *associated to* $\mathcal{S} = (A_i, B_i, d_i)_{i\in\{1,\ldots,n\}}$, *compute either*

- *an update of* $\mathcal{D}(\mathcal{S})$ *by adding* $(A_{n+1}, B_{n+1}, d_{n+1})$, *thus obtaining* $\mathcal{D}(\mathcal{S}')$,
- *an update of* $\mathcal{D}(\mathcal{S})$ *by withdrawing* (A_n, B_n, d_n), *thus obtaining* $\mathcal{D}(\mathcal{S}'')$, *where* $\mathcal{S}'' = (A_i, B_i, d_i)_{i\in\{1,\ldots,n-1\}}$,
- *the maximum load*

$$\max\left\{\sum_{i\in I} d_i \text{ such that } I \subseteq \{1,\ldots,n\} \text{ and } \bigcap_{i\in I, t_i^{init}\leq t<t_i^{end}} rect(A_iA_iB_i) \neq \emptyset\right\}.$$

The complexity of the problem is given by the maximum complexity of the three operations, multiplied per n.

***Problem* 7** (SAT CAC ACCESS) *Given a set of communication requests* $\mathcal{S}' = (A_i, d_i)_{i\in\{1,\ldots,n+1\}}$, d_i *being a weight, and a data structure* $\mathcal{D}(\mathcal{S})$ *associated to* $\mathcal{S} = (A_i, d_i)_{i\in\{1,\ldots,n\}}$, *compute either*

- *an update of* $\mathcal{D}(\mathcal{S})$ *by adding* (A_{n+1}, d_{n+1}), *thus obtaining* $\mathcal{D}(\mathcal{S}')$,
- *an update of* $\mathcal{D}(\mathcal{S})$ *by withdrawing* (A_n, d_n), *thus obtaining* $\mathcal{D}(\mathcal{S}'')$, *where* $\mathcal{S}'' = (A_i, d_i)_{i\in\{1,\ldots,n-1\}}$,
- *the maximum load*

$$\max\left\{\sum_{i\in I} d_i \text{ such that } I \subseteq \{1,\ldots,n\} \text{ and } \bigcap_{i\in I} sat(A_i) \neq \emptyset\right\}.$$

The complexity of the problem is given by the maximum complexity of the three operations, multiplied per n.

3.2 Intersection of Rectangles: the Max Depth Problem

Coming back to some more natural computational issue, we can formulate the Lee's measure problem. Its d-dimensional version in the Euclidian space is as follows.

***Problem* 8** *Let $v_1, \ldots, v_d$ be a series of independent vectors in $\mathbb{R}^d$, and $R^1, \ldots, R^n$ be a series of regular hypercubes with respect to $v_1, \ldots, v_d$.*
What is the measure of the area $\bigcup_{j \in \{1,\ldots,n\}} R^j$?

Figure 7 illustrates an instance of the Klee's measure problem in two dimensions. The output is the measure of the dashed area.

Interestingly, all the solutions we mentioned for the dynamic depth problem [10, 14, 2] where in fact introduced to solve Klee's measure problem. An in-between version, the DEPTH problem, can be defined as follows:

***Problem* 9** ($1 - \text{DEPTH}^d$) *Let $v_1, \ldots, v_d$ be a series of independent vectors in $\mathbb{R}^d$, and $R^1, \ldots, R^n$ be a series of regular hypercubes with respect to $v_1, \ldots, v_d$.*
What is the largest set $I \subseteq \{1, \ldots, n\}$ such that $\bigcap_{j \in I} R^j \neq \emptyset$?

Also the weighted versions can be given as follows:

***Problem* 10** (DEPTH^d) *Let $v_1, \ldots, v_d$ be a series of independent vectors in $\mathbb{R}^d$, and $R^1, \ldots, R^n$ be a series of regular hypercubes with respect to $v_1, \ldots, v_d$. Let $w_1, \ldots, w_n$ be weights assigned to the hypercubes.*
What is the set $I \subseteq \{1, \ldots, n\}$ with maximum $\sum_{j \in I} w_j$ such that $\bigcap_{j \in I} R^j \neq \emptyset$?

In the sequel we will investigate reductions among those problems. In particular, we will first prove that any 'Access' problem is equivalent to the corresponding 'Link' version. Moreover, we will give some strong evidence that the latter two inequality are "strict", i.e. the 'Cac' problems are harder than the 'Tot' ones.

3.3 Reductions Among the Problems

We will show that having one or even both fixed sides does not make the problem easier. Additionally, the reductions essentially group the problems according to their static/dynamic nature.

The following proposition can be easily proved by considering the $d+1$-th dimension as the "time" dimension of a dynamic version of the same problem in d dimensions.

Proposition 1 $\text{DEPTH}^{d+1} \leq \text{DYNAMIC DEPTH}^d$

Theorem 6
- $\text{DEPTH}^2 \sim \text{SAT TOT LINKS} \sim \text{SAT TOT ACCESS}$
- $\text{DEPTH}^3 \leq \text{DYNAMIC DEPTH}^2 \sim \text{SAT CAC LINKS} \sim \text{SAT CAC ACCESS}$

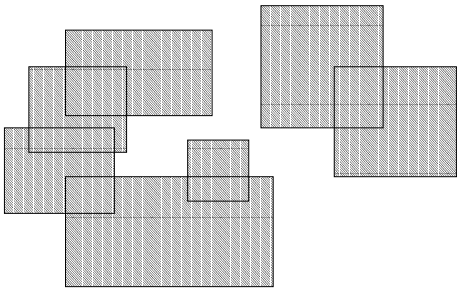

Figure 7: Klee's measure problem in 2D.

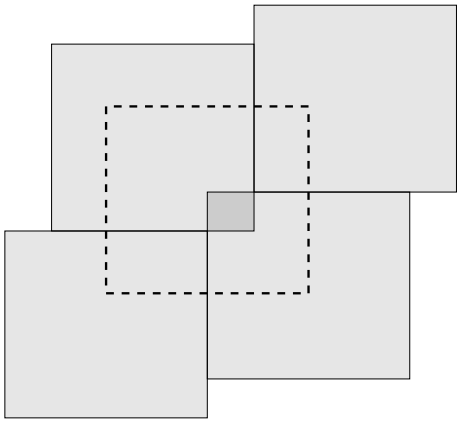

Figure 8: Drawing a rectangle of arbitrary dimensions inside a box with four squares of identical dimensions.

PROOF.

Firstly, let us consider the first part of the theorem. Obviously we have SAT TOT ACCESS $\leq$ SAT TOT LINKS $\leq$ DEPTH^2. Note that a rectangle of arbitrary dimensions can be seen as a superprint of four squares of equal dimensions, as shown in Figure 8. This proves SAT TOT ACCESS $\leq$ DEPTH^2 and the first part of the theorem.

Concerning the second part, the first inequality follows from Proposition 1. As for the remaining reductions, we simply observe that the above transformations can also be applied to the dynamic version of these problems. Indeed, each insertion/widraw operation in one problem translates into a constant number of insertion/widraw operations of the other one. Hence the theorem follows. □

By applying the algorithms given in [10] and [14, 2], we obtain the following upper bounds:

Theorem 7 *Let be* STATIC $\in$ {SAT TOT LINKS, SAT TOT ACCESS, DEPTH^2} *and let be* DYNAMIC $\in$ {SAT CAC LINKS, SAT CAC ACCESS, DYNAMIC DEPTH^2}. *Then, the following two facts hold:*

- $M -$ STATIC $\leq n \log(n) \log(nM)$
- $M -$ DYNAMIC $\leq n^{3/2} \log(n) \log(nM)$

Finally, the following result is a consequence of Corollary 2 and of Theorem 6.

Corollary 3 *Let be* DYNAMIC $\in$ {SAT CAC LINKS, SAT CAC ACCESS, DYNAMIC DEPTH^2}. *Then, the following bounds hold:*

$$n^{3/2} \leq \# - \text{DYNAMIC} \leq n^{3/2} \log n.$$

4 Conclusions and Open Problems

In this paper, we have seen several aspects of the complexity of Klee measure problems and satellite communications. Under the "funny" assumption, we can even state tight bounds. The next step would be of course to remove this assumption. This is related to the complexity of the non-funny matrix multiplication - and in particular to the classical matrix multiplication - which is known to be under $O(N^{2.376})$ [3], and obviously at least $\Omega(N^2)$. Improving significantly this bound, however, seems to be difficult [1]. As a result, how far we are form the optimum in our case is even more widely open. A natural question is also whether we could use a matrix-multiplication in order to compute either DEPTH^3 or even DYNAMIC DEPTH^2 more efficiently. The ideal would be to reverse the reduction from matrix-multiplication to our problems. This would provide a more clear view of our concerns.

References

[1] M. Bläser. A - lower bound for the rank of - matrix multiplication over arbitrary fields. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999.

[2] B.S. Chlebus. On the Klee's measure problem in small dimensions. In *SOFSEM-98*, LNCS 1521, pages 304–311. Springer-Verlag, 1998.

[3] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. *Journal of Symbolic Computation*, 9:251–280, 1990.

[4] F. J. Dietrich. The globalstar satellite cellular communication system design and status. In *Proceedings of the fifth International Mobile Satellite Conference*, pages 139–144, Pasadena, CA, June 1997.

[5] M. Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5:49–60, 1976.

[6] Z. Galil and O. Margalit. All pairs shortest distances for graphs with small integer length edges. *Information and Computation*, 134:103–139, 1997.

[7] Z. Galil and O. Margalit. All pairs shortest paths for graphs with small integer length edges. *Journal of Computer and System Sciences*, 54:243–254, 1997.

[8] J. Galtier. Geographical reservation for guaranteed handover and routing in low earth orbit constellations. *Telecommunication Systems*, 2000. To appear, http://www.prism.uvsq.fr/public/galtier/geores.pdf.

[9] Y. C. Hubbel and L. M. Sanders. A comparison of the IRIDIUM and AMPS systems. *IEEE network*, March/April 1997.

[10] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4:310–323, 1983.

[11] L.R. Kerr. *The Effect of Algebraic Structure on the Computational Complexity of Matrix Multiplication*. PhD thesis, Cornell University, 1970.

[12] D. M. Kohn. Providing global broadband internet access using low-earth-orbit satellites. *Computer networks and ISDN systems*, 29:1763–1768, 1997. A technical summary of the system by the marketing manager of Teledesic corporation.

[13] R. J. Leopold. Low-earth orbit global cellular communications network. In *Proceedings of ICC '91*, pages 1108–1111, 1991.

[14] M.H. Overmars and C.K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):550–556, 1991. Also in *FOCS-88*.

[15] R. Seidel. On the all-pairs-shortest-path problem in undirected unweighted graphs. In *Journal of Computer and System Sciences*, volume 51, 1995. Also in *STOC-92*.

[16] M. A. Sturza. Architecture of the teledesic satellite system. In *Proceedings of the International Mobile Satellite Conference '95*, pages 214–218, Ottawa, 1995. A summary of much of the information presented in the Calling paper, with a description of the final 840 active-satellite design, subsystems and communications payloads.

[17] T. Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43:195–199, 1992.

[18] T. Takaoka. Subcubic cost algorithms for the all pairs shortest path problem. *Algorithmica*, 20:309–318, 1998.

[19] U. Zwick. All pairs shortest paths in weighted directed graphs - exact and almost exact algorithms. In *FOCS-98*, pages 310–319, 1998.

A Proof of Theorems 1-2

PROOF. Suppose for instance that

$$C_{inter} < \max \left\{ |I| \text{ such that } I \subseteq \{1, \ldots, p\} \text{ and } \bigcap_{i \in I} rect(A_i A_i B_i) \neq \emptyset \right\}$$

and take an I that achieves the maximum. Then consider a point P in

$$S = \bigcap_{i \in I} rect(A_i A_i B_i),$$

and consider the instant when a center c_l of a link l is on P. Clearly, the demand required on l is superior to its capacity. □

The proof of Th. 2 is almost identical to the previous one.

B The Proof of Theorem 4

PROOF. We first observe that, according to the definition of fixed program, the sequence of max and + operations performed at run-time by Alg *does not depends on the weights.* So, such a sequence is completely determined by the rest of the input, that is the shapes of the hypercubes. So, fix a set $\mathcal{R} = \{R_1, \ldots, R_n\}$, as defined in Problem 3. Thus, by expanding the computations performed by the loop instructions into a sequence of fixed program instructions, we easily obtain a fixed program that solves $\#-\text{DYNAMIC DEPTH}^{d,\mathcal{R}}$. Moreover, its length is at most the number of operations performed by Alg. This proves the theorem. □

Contents

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399